
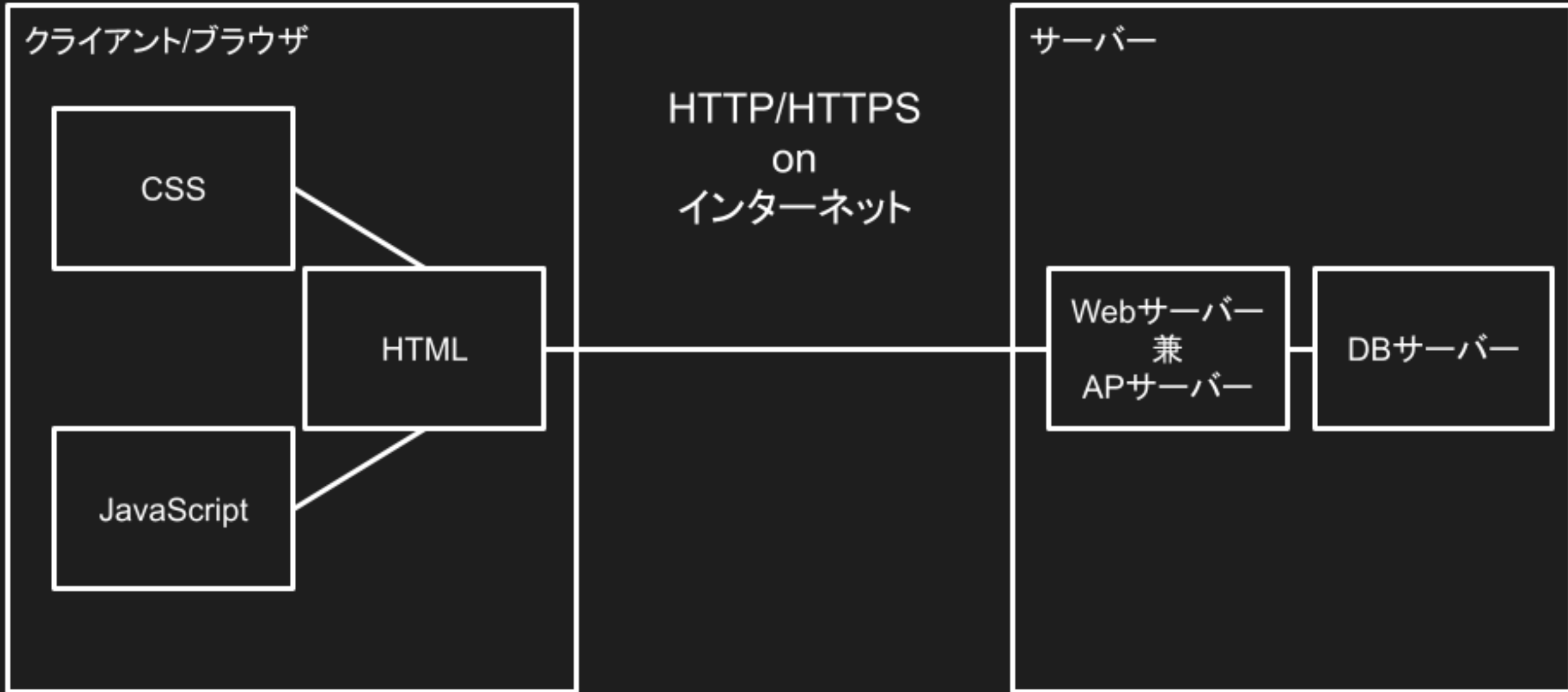


Webアプリケーション入門 🚀

DOMってなに？

はじめに：今日やること

- 今日のゴール: 
 - DOMとは何かを知る
 - DOMを使ってHTMLを操作する方法を知る



Webページってどうできてる？

- **HTML:** Webページの骨組みや内容を書く言語
 - 「ここにタイトル」「ここに文章」「ここに画像」
- **CSS:** 見た目を飾る言語
 - 「文字の色は赤」「背景は青」「幅はこのくらい」
- **JavaScript:** ページに動きをつける言語
 - 「ボタンを押したら何かする」「時間で表示を変える」

HTMLだけだと、動きのあるページをつくれない…

動きのあるページはどう作る？

そこで **JavaScript** の出番！ ✨

でも、JavaScriptはどうやってHTMLの特定の場所を操作するの？

HTMLファイル（ただのテキスト）を直接いじるのは大変… 🤔

そこで登場！ DOM（ドム）！

DOM = Document Object Model

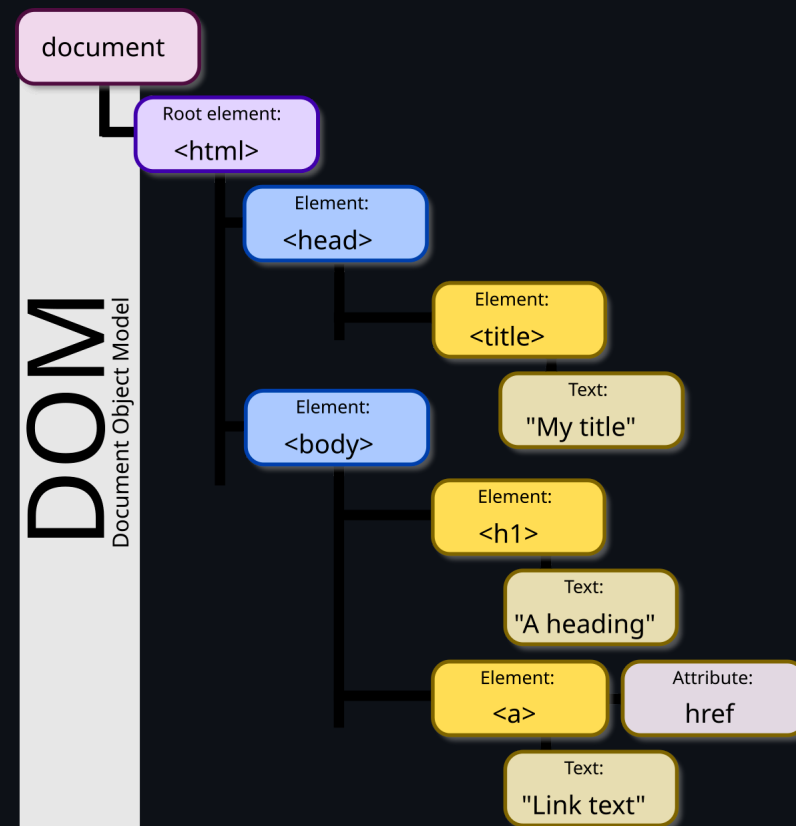
一言でいうと…

ブラウザがHTMLを読み込んで作る

「ページの構造図」

+

JavaScriptがその構造図を操作するための「ルール」



なぜDOMが必要なの？

- 元のHTMLファイルは、ただの文字の集まり（テキスト）。
- JavaScriptが「あの見出しを変えたい！」と思っても、テキストの中から探すのは難しい。

そこで！

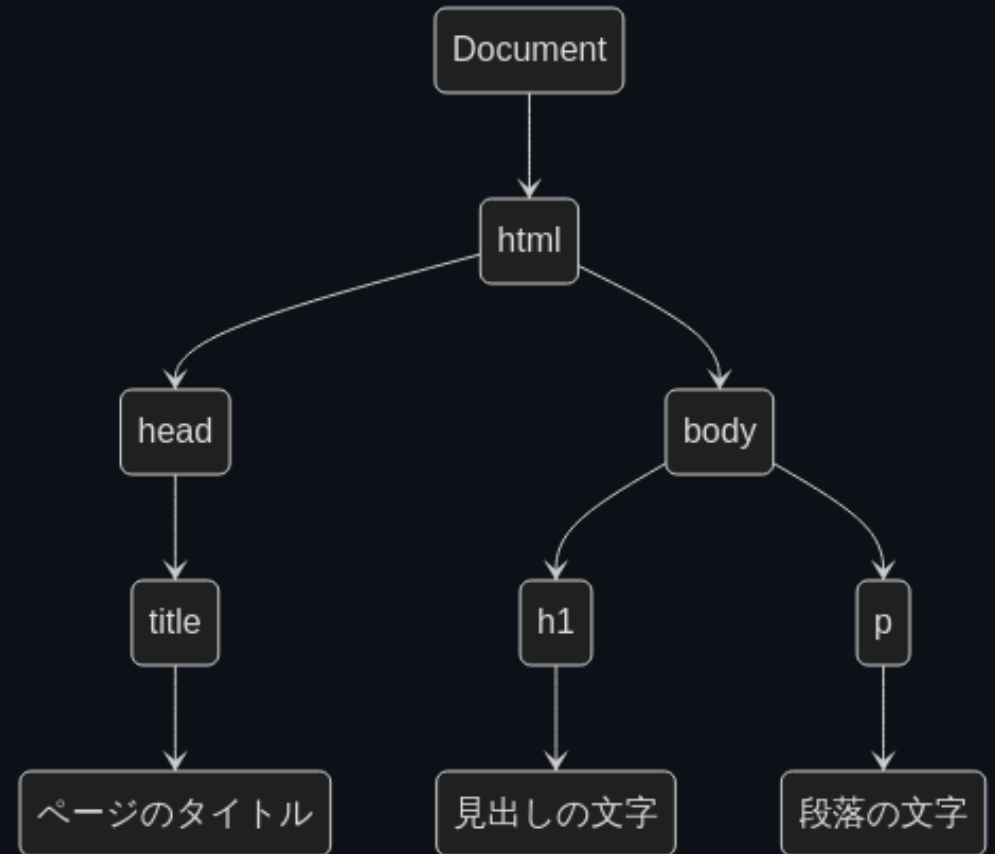
ブラウザがHTMLを読み込む時に、理解しやすい**「整理された形（＝DOM）」**を内部に作ってくれる！

DOMのイメージ：ページの 家系図 🌳

ブラウザはHTMLを読み込むと、こんな感じのツリー構造（家系図みたいなもの）を頭の中に作ります。

- 一番上に `<html>`（ご先祖様）
- その下に `<head>` や `<body>`（子供）
- さらに下に `<h1>` や `<p>`（孫）...

この一つ一つが**操作できる部品（オブジェクト）**になっています。これが **DOM** です。



JavaScriptは何をするの？ (DOM操作)

JavaScriptは、ブラウザが作ったこの「**構造図 (DOM)**」を見て、ページを操作します。

1. **探す**: 「IDが `main-title` の見出し部品を探して！」
2. **中身を変える**: 「その見出しの文字を『ようこそ！』に変えて！」
3. **見た目を変える**: 「IDが `info` の段落の文字色を赤にして！」
4. **追加する**: 「このリストの最後に新しい項目を追加して！」
5. **削除する**: 「この画像部品はもういらなから消して！」

具体的な動き (イメージ)

ユーザーがボタンをクリック！



JavaScriptが検知！



JavaScript: 「ブラウザさん、DOMの中からIDが `message` の部品を探して、その中の文字を『クリックされました！』に変えてください！」



ブラウザ: (DOMを操作)



画面の表示が変わる！ ✨

まとめ

- **DOM** は、ブラウザがHTMLを解析して作る**ページの構造図（+操作ルール）**。
- JavaScriptは、この**DOMを操作すること**で、Webページの内容や見た目を**動的に変更**できる。
- ボタンを押したら表示が変わるような**インタラクティブなWebページ**を作るための基本中の基本！

準備：HTMLファイル (index.html)

まずは、操作の対象となる簡単なHTMLを用意します。

ポイント：各要素に `id` をつけておく！

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM基本</title>
</head>
<body>
  <h1 id="main-title">こんにちは！</h1>
  <p id="text-area">ここにメッセージが...</p>
  <button id="change-button">メッセージ変更</button>

  <script src="script.js"></script>
</body>
</html>
```

1. 探す (JavaScript: script.js)

`document.getElementById('ID名')` で要素を見つけます。

```
// IDを使ってHTML要素を探して、変数に入れる
const titleElement = document.getElementById('main-title');
const textElement = document.getElementById('text-area');
const buttonElement = document.getElementById('change-button');

// ↓見つけた要素が変数に入る
// titleElement, textElement, buttonElement
```

- `getElementById` は「指定したIDの部品を探して！」という命令。
- `const 変数名 = ...` で、見つけた部品を名前をつけて保管。

2&3. 反応して変える (JS: script.js 続き)

「ボタンがクリックされたら処理を実行する」ように設定します。

➡ `addEventListener` を使います。

```
// buttonElementに対して「'click'があったら実行して」と命令
buttonElement.addEventListener('click', function() {
    // ここにクリックされた時の処理を書く！
    // (次のスライドで解説)
});
```

- `buttonElement` : どの要素が？ (さっき見つけたボタン)
- `'click'` : いつ？ (クリックされたら)
- `function() { ... }` : 何をする？ (この `{ }` の中の処理)

クリックされた時の処理の中身 (JS)

`addEventListener` の `{ }` の中に、実行したい処理を書きます。

```
buttonElement.addEventListener('click', function() {  
  // --- ここから処理 ---  
  
  // p要素 (textElement) の文字を変更  
  textElement.textContent = 'クリックされました！';  
  
  // h1要素 (titleElement) の文字色を赤に  
  titleElement.style.color = 'red';  
  
  // ボタン自身の文字も変更  
  buttonElement.textContent = '変更完了！';  
});
```

- `textContent` : 要素の**文字**を変える
- `style.プロパティ名` : 要素の**見た目 (CSS)** を変える

大事なポイント

- `<script>` タグの位置:
基本は `</body>` の直前に書く。HTML要素を読み込んだ後にJSが実行されるようにするため。
- ID はユニークに:
`id="xxx"` は、1つのHTMLページ内で同じ名前を使わないようにする。（名札は一人一枚！）
- 開発者ツール (F12):
`console.log()` の結果を見たり、エラーを確認したりできる超便利ツール！必ず使い方を覚えよう。

まとめ

- JavaScriptは **DOM (設計図)** を使ってHTMLを操作する。
- 基本は「**探す** → **反応する** → **変える**」
- `getElementById` で探す。
- `addEventListener` でクリックなどに反応させる。
- `textContent` , `style` で変える。

まずは簡単なことから試してみよう！ 💪